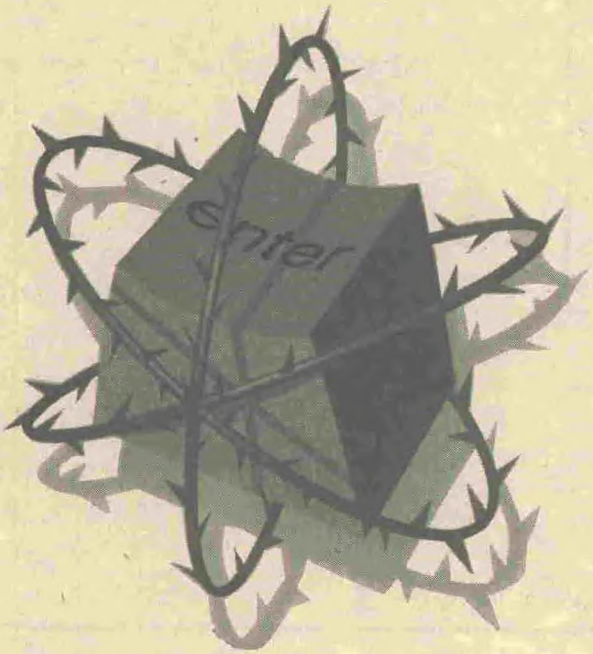
ณรงค์ จันทโพธิ์

SOFT MAIL

สร้างสรรค์การใช้งานเม้าส์ด้วย OOP : การสร้างปุ่มกดแบบรูปภาพ



การสร้างปุ่มกดแบบรูปภาพ ซึ่งจะมีลักษณะการทำงานคล้ายกับปุ่มกดแบบข้อความ

สำหรับบทความตอนนี้เราจะมาพูดถึงเรื่องการสร้างปุ่มกดแบบรูปภาพ ลักษณะการทำงานโดยทั่วไปจะคล้ายกับปุ่มกดแบบข้อความซึ่งผมได้นำเสนอไปแล้วในฉบับที่ 133 ปุ่มกดแบบรูปภาพนี้ จะใช้รูปแบบบิตแม็พที่ได้จากโปรแกรมวาดรูปที่ผมได้เสนอไปในฉบับที่แล้ว รูปภาพจะมีขนาด 24x24 พิกเซล แนวทางในการออกแบบออบเจ็กต์นี้ผมจะนำเสนอการออกแบบเป็น 2 แนวทางดังนี้ครับ

1. โดยการได้รับการสืบทอดมาจากออบเจ็กต์ PushButton
2. โดยการได้รับการสืบทอดมาจากออบเจ็กต์ TextButton

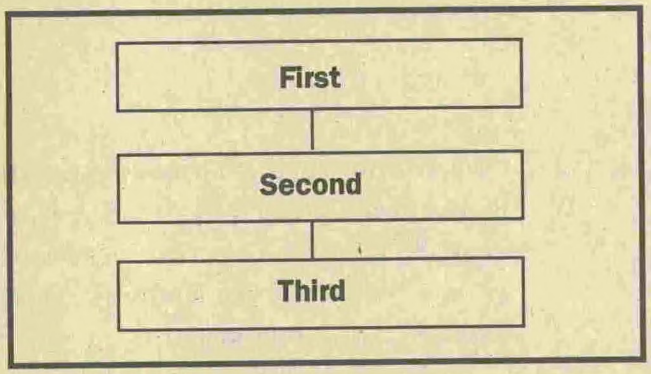
ในวิธีการแบบแรกนั้นการออกแบบจะเหมือนกับการออกแบบออบเจ็กต์ที่ผ่าน ๆ มา จากข้างต้นที่ผมกล่าวว่าลักษณะการทำงานเหมือนกับปุ่มกดแบบข้อความ จะต่างกันตรงเพียงหน้าตา นั่นหมายถึงว่า Method ต่าง ๆ ที่จะสร้างขึ้นมาก็จะมีการทำงานเหมือนกับปุ่มกดแบบรูปภาพ ส่วนที่จะต้องทำการออกแบบใหม่คือ Method ที่ทำหน้าที่ในการวาดรูปปุ่มกดบนหน้าจอ

สำหรับวิธีการแบบที่ 2 นั้นเราจะมาพิจารณาการทำงานของปุ่มกดทั้งสองว่ามีส่วนการทำงานที่เหมือนกันนั่นหมายความว่าเรามีส่วนการทำงานบางส่วนอยู่แล้ว เพียงแต่เราสร้างหรือปรับปรุงส่วนการทำงานบางส่วนที่มีอยู่ให้สามารถควบคุมปุ่มกดแบบรูปภาพให้ทำงานได้เพียงเท่านั้นก็เป็นการเพียงพอ สำหรับวิธีการนี้จะเหมาะสมกับการเขียนโปรแกรมแบบ OOP คือการนำโค้ดเก่ามาปรับปรุงใหม่โดยอาจมีการเพิ่มคุณสมบัติใหม่เข้าไปแล้วสร้างเป็นออบเจ็กต์ใหม่ขึ้นมา (Derived class) แต่ก่อนที่จะมาออกแบบออบเจ็กต์ปุ่มกดแบบรูปภาพเราจะมาทำความเข้าใจบางประการในการสืบทอดคุณสมบัติจาก Class หนึ่งไปยัง Class ที่จะสร้างขึ้นใหม่ จากตอนที่ผ่านมาการสืบทอดคุณสมบัติของ Class บรรพบุรุษไปยัง Class ลูกนั้นไม่มีความซับซ้อนหรือยุ่งยากใด ๆ แต่ในครั้งนี้จะมีความยุ่งยากเพิ่มขึ้นตรงที่ตัว Class บรรพบุรุษนั้นมี Constructor ที่มีการส่ง Argument ด้วย ตรงจุดนี้เองที่เราจะต้องมาทำความเข้าใจกับการสร้างออบเจ็กต์ใหม่จากบรรพบุรุษที่ Constructor มี Argument ด้วย ดังในหัวข้อถัดไปครับ

ลำดับการทำงาน Constructor และ Destructor

โดยปกติแล้ว Constructor และ Destructor จะไม่สามารถสืบทอดต่อไปยัง Class ลูกได้ ในขั้นตอนของการคอมพายล์ Constructor จะถูกเรียกขึ้นมาโดยอัตโนมัติ แต่หลังจากนั้นจะเป็นหน้าที่การทำงานของ Class ลูกต่อ

ไป เช่นเดียวกัน Destructor จะถูกเรียกขึ้นมาทำงานโดยอัตโนมัติเมื่อโปรแกรมทำงานนอก Scope ของออบเจ็กต์นั้น ๆ ตรงจุดนี้เองที่เราจะต้องทำความเข้าใจในลำดับการทำงานของ Constructor และ Destructor เมื่อออบเจ็กต์ลูกเริ่มต้นทำงานจนสิ้นสุดการทำงานซึ่งในตอนที่ผ่าน ๆ มาผมยังไม่ได้กล่าวถึงเลย



รูปที่ 1 ตัวอย่างการสืบทอดของ Class 3 Class

เราจะมาพิจารณาลำดับการทำงานของ Constructor กันก่อนครับ จากรูปที่ 1 แสดงถึงลำดับการสืบทอดจากบนลงล่าง โดยผมกำหนดว่าทุกออบเจ็กต์มี Constructor ด้วย ลำดับการทำงานของ Constructor เริ่มเกิดขึ้นทันทีที่มีการประกาศ Class ลูกโดยมีลำดับดังนี้

```
First::First()
Second::Second()
Third::Third()
```

และสำหรับลำดับการขึ้นมาทำงานของ Destructor ก็จะกลับกันกับ Constructor นั่นคือ

```
Third::~Third()
Second::~Second()
First::~First()
```

ส่วนต่อไปที่เราจะมาพิจารณากันตรงเงื่อนไขที่ว่า Constructor ของ Class พ่อนั้น มีการส่งผ่าน Argument ด้วย จากที่กล่าวไปแล้วว่า Constructor ของ Class พ่อนั้นจะถูกเรียกขึ้นมาทำงานในขั้นตอนการคอมพายล์ ตรงจุดนี้เองที่จะต้องทำการระบุอะไรสักอย่างต่อ Constructor ของ Class พ่อในขณะทำการประกาศสร้าง Class ลูก โค้ดในลิสติงที่ 1 ต่อไปนี้ผมสมมติขึ้นมาตามผังการสืบทอดของทั้ง 3 Class

ลิสติงที่ 1 DERIVED.CPP ตัวอย่างของการสืบทอดของ Class 3 Class

```
1:  #include<iostream.h>
2: #include<conio.h>
3: class Frist{
4: public:
5: private:
6: int a,b;
7: protected: //Data for derived class only
8:     Frist(int,int);
9:     ~Frist();
10: };
11: //*************************************************************************
12: class Second:public Frist{
13: public:
14: private:
15: char c;
16: protected: //Data for derived class only
17:     Second(int,int,char);
18:     ~Second();
19: };
20: //*************************************************************************
21: class Third:public Second{
22: public:     //For every where you need to invoke
23:     Third(int,int,char,float);
24:     ~Third();
25: private:
26: float d;    //You can't access with command: Third.d
27: protected:
28: };
29: //*************************************************************************
30: Frist::Frist(int a1,int b1){
31:     a = a1;
32:     b = b1;                                                          →
```

ลิสติงที่ 1 (ต่อ)

```
33:     cout << "In Frist a = "<< a << " b = "<< b << "\n";
34: }
35: //***************************************************************************
36: Frist::~Frist(){
37:     cout << "Good bye from Frist\n";
38: }
39: //***************************************************************************
40: Second::Second(int a1,int b1,char c1):Frist(a1,b1){
41:     c = c1;
42:     cout << "In Second c = "<< c <<"\n";
43: }
44: //***************************************************************************
45: Second::~Second(){
46:     cout << "Good bye from Second\n";
47: }
48: //***************************************************************************
49: Third::Third(int a1,int b1,char c1,float d1):Second(a1,b1,c1){
50:     d = d1;
51:     cout << "In Third d = "<< d <<"\n";
52: }
53: //***************************************************************************
54: Third::~Third(){
55:     cout << "Good bye from Third\n";
56: }
57: //***************************************************************************
58: main(){
59: Third Obj(10,20,'a',20.5);
60:     getch();
61:     return 0;
62: }
```

ในแต่ละ Class นั้นพอจะแบ่งส่วนของ Property เป็น 3 ส่วน ซึ่งการที่มีแต่ละส่วนเพื่อกำหนดว่าใครบ้างที่จะสามารถใช้งาน Data เหล่านั้นได้บ้าง ผมก็ขอทบทวนดังนี้ครับ

- **public** ข้อมูลที่ประกาศในส่วนนี้จะสามารถถูกใช้ได้จากทุก ๆ ตำแหน่งในโปรแกรม

- **protected** ข้อมูลที่ประกาศในส่วนนี้จะสามารถถูกใช้ได้จาก Class ที่ได้รับการสืบทอดจาก Class นี้เท่านั้น

- **private** ข้อมูลที่ประกาศในส่วนนี้จะสามารถถูกใช้ได้เฉพาะภายใน Class นั้น ๆ เท่านั้น

สำหรับที่ตัว Constructor ของ Class Second จะมีการสร้างส่วนหัวเป็น

```
Second::Second(int a1,int b1,char
c1):Frist(a1,b1)
```

ในส่วนท้ายเป็นการส่งข้อมูลไปยัง Constructor ของ First ซึ่งข้อมูลที่ส่งผ่านไปนี้จะต้องมีตัวตนอยู่จริง ในที่นี้จะเห็นว่า a1 และ b1 เป็น Argunment ที่ได้รับมาจาก Second อีกทอดหนึ่ง เช่นเดียวกันที่ตัว Constructor ของ Class Third จะมีการสร้างส่วนหัวเป็น

```
Third::Third(int a1,int b1,char c1,float
d1):Second(a1,b1,c1)
```

ข้อมูลที่ส่งไปให้กับ Constructor ของ Second โดย Argument ที่ส่งไปนั้นก็ได้มาจาก Third อีกทอดหนึ่งเช่นกัน ส่วนที่ Destructor ผมได้เคยกล่าวไปแล้วถึงคุณลักษณะของมันข้อหนึ่งที่ว่า Destructor นั้นจะไม่สามารถมีการรับ-ส่ง Argument ได้ แต่เราสามารถจะเขียนคำสั่งใด ๆ เข้าไปใน Destructor ได้ โดยจะใช้งานมันเมื่อจะจบหน้าที่ของออบเจ็กต์นั้น ๆ เช่นการคืนหน่วยความจำให้แก่ระบบเป็นต้น สำหรับในตัวอย่างนี้ผมแสดงให้เห็นถึงลำดับการทำงานของมันผมจึงกำหนดคำสั่งแสดงผลบนหน้าจอเมื่อมันถูกเรียกให้ทำงาน ตัวโปรแกรมนี้จะแสดงถึงลำดับการทำงานของ Constructor และ Destructor คือเมื่อโปรแกรมทำงานจะปรากฏข้อความบนหน้าจอ 3 บรรทัดตามลำดับดังนี้

```
In First a = 10 b = 20
In Second c = a
In Third d = 20.5
```

จะเห็นว่าลำดับการทำงานของ Constructor ทำงานดังที่ได้กล่าวไปแล้ว และเมื่อกดคีย์ใด ๆ ก็จะปรากฏข้อความอีก 3 บรรทัดดังนี้ครับ

```
Good bye from Third
Good bye from Second
Good bye from First
```

ทั้ง 3 บรรทัดนี้เกิดจากการทำงานของ Destructor คือเมื่อจบโปรแกรมจะเป็นการออกนอก Scope ของ Obj โดยทันที ผลทำให้ Deststructor ถูกเรียกทำงาน ซึ่งลำดับก็จะตรงกันข้ามกับ Constructor ดังที่ผมได้กล่าวไปแล้วครับ

โดยปกติแล้วถึงแม้ว่าเราจะไม่มีการสร้าง Constructor หรือ Destructor ในออบเจ็กต์ใด ๆ ก็ตาม แต่โดย Default แล้วตัว Constructor และ Destructor นี้จะถูกสร้างขึ้นมาเสมอโดยที่หน้าที่หลักของ Constructor คือการจองพื้นที่หน่วยความจำให้กับตัวออบเจ็กต์ ส่วนหน้าที่หลักของ Destructor ก็จะเป็นการคืนพื้นที่หน่วยความจำที่ถูกจองโดย Constructor ให้แก่ระบบ นอกจากนี้คุณสมบัติอีกประการที่น่าจะกล่าวถึงคือการ Overloading Constructor คือการสร้าง Constructor ตัวใหม่ขึ้นมาโดยที่เรามีการสร้าง Constructor ไว้แล้ว ในการ Overloading Consturctor นี้ก็เป็นไปตามกฎการ Overloading พวกฟังก์ชันธรรมดา (Constructor ก็เปรียบเหมือนฟังก์ชันธรรมดานั่นเองเพียงแต่ไม่สามารถ return ค่าใด ๆ ได้) จากตัวอย่างในลิสติงที่ 1 หากท่านไม่ต้องการสร้าง Obj ที่มีการส่ง Argument ก็อาจทำการ Overloading โดยการสร้าง Constructor อีกตัวที่ไม่มีการส่ง Argument ใด ๆ ซึ่ง Constructor ที่ไม่มี Argument นี้ ใน C++ จะมีชื่อเรียกว่า Default Constructor ในตัว Constructor ก็จะต้องมีการส่งค่าต่าง ๆ ไปให้กับ Constructor ของ Second เช่นเดิม เพียงแต่ค่าที่ส่งต้องมีตัวตนจริง เช่น อาจเป็นตัวแปรที่ประกาศขึ้นใน Constructor ของ Third หรือเป็นค่าคงที่ใด ๆ ดังเช่น

```
Third::Third():Second(10,20,'a')
```

ที่จริงแล้วเรื่องของ Constructor ยังมีอีกมากซึ่งผมคงยกให้ท่านผู้อ่านที่สนใจในเรื่องของ OOP ไปค้นคว้าต่อแล้วกันครับ เป็นต้นว่าการรับการสืบทอดจาก Class 2 ตัวพร้อม ๆ กันเราจะพิจารณาลำดับการทำงานของ Constructor อย่างไร และมีการประกาศอย่างไรเป็นต้น

ลิสติงที่ 2
การประกาศ Opject PicButton โดยการสืบทอดคุณสมบัติจาก Button

```
class PicButton:public PushButton{
   1: public:
   2:    PicButton(int,int,char *,int,unsigned
int,int);
   3: protected:
   4:    virtual void Destroyed(){}
   5:    virtual void Draw();
   6:    virtual void DrawButtUp();
   7:    virtual void DrawButtDown();
   8:    virtual int Pressed();
   9:    virtual int GetButtRelease();
   10:    virtual void DrawFocus(int );
   11: };
```

ลิสติงที่ 3 PICBTN.CPP โดยการสืบทอดคุณสมบัติจาก Button

```
1:    #include<graphics.h>
2:    #include"mouobj.h"
3:    #include"stdfon.h"
4:    #include"objects.h"
5:    extern unsigned int MouEvent;
6:    extern int MouX,MouY;
7:    //*************************************************************************
8:    void DrawPic(int x,int y,char *Pic){
9:    int maxc,maxr,col,row;
10:        maxr = *Pic;
11:        Pic++;
12:        maxc = *Pic;
13:        Pic++;
14:        HideMouse();
15:        for(row=y;row<maxr+y;row++)
16:           for(col=x;col<maxc+x;col++){
17:   if(*Pic>=0){
18:      putpixel(col,row,*Pic);
19:      }
20:   Pic++;
21:   }
22:        ShowMouse();
23:    }
24:    //*************************************************************************
25:    PicButton::PicButton(int Col,int Row,char *title,int style,unsigned int cmd,int BMsg){
                                                                              →
```

ลิสติงที่ 3 (ต่อ)

```
26:       SCol  = Col;
27:       SRow  = Row;
28:       ECol  = SCol + 28;  //Fixed button width
29:       ERow  = SRow + 28;  //Fixed button hieght
30:       Style = style;
31:       Cmd   = cmd;
32:       Title = title;      //point to picture address
33:       ObjType = PICBUTTON;
34:       RetCmd  = BMsg;
35:       if(Style&VISIBLE)
36:          DrawButtUp();
37:    }
38:    //*************************************************************************
39:    void PicButton::Draw(){
40:       Style|=VISIBLE;
41:       DrawButtUp();
42:    }
43:    //*************************************************************************
44:    void PicButton::DrawFocus(int Down){
45:       setlinestyle(4,0xAAAA,1);
46:       setcolor(BLACK);
47:       rectangle(SCol+3,SRow+3,ECol-3,ERow-3);
48:       setlinestyle(0,0,1);
49:    }
50:    //*************************************************************************
51:    void PicButton::DrawButtUp(){
52:       HideMouse();
53:       PushButton::DrawButtUp();
54:       DrawPic(SCol+2,SRow+3,Title);
55:       if(Style&FOCUSED){
56:          DrawFocus(0);
57:          }
58:       ShowMouse();
59:    }
60:    //*************************************************************************
61:    void PicButton::DrawButtDown(){
62:       HideMouse();
63:       PushButton::DrawButtDown();
64:       DrawPic(SCol+3,SRow+4,Title);
65:       DrawFocus(1);
66:       ShowMouse();
67:    }
68:    //*************************************************************************
69:    int PicButton::GetButtRelease(){
70:       int Active = 3;
71:          while(Active==3){
72:  GetMouInfo(&MouEvent,&MouX,&MouY);
73:  if((MouEvent==NO_DOWN)&&
74:     MouInBox(MouX,MouY,SCol,SRow,ECol,ERow)){
75:     Active = 1;
76:     }
77:  else
78:     if((MouEvent==L_DOWN)&&
79:     (!MouInBox(MouX,MouY,SCol,SRow,ECol,ERow)))
80:     Active = 0;
81:  }
82:          DrawButtUp();
83:       return Active;
84:    }
85:    //*************************************************************************
86:    int PicButton::Pressed(){
87:          if((MouX)<-1)           //Reference key was pressed
88:  return 1;
89:          if((MouEvent==L_DOWN)&&
90:  (MouInBox(MouX,MouY,SCol,SRow,ECol,ERow))){
91:  DrawButtDown();
92:  return 1;
93:  }
94:          else
95:  return 0;
96:    }
```

เริ่มต้นสร้าง

ตัวอย่างแรกในลิสติงที่ 2 และ 3 เป็นการสืบทอดจาก Class Button มีฟังก์ชัน DrawPic() ซึ่งเป็นฟังก์ชันใหม่ที่สร้างขึ้นมาเพื่อใช้ในการวาดรูปที่ถูกเรียกขึ้นมาทำงานภายในออบเจ็กต์ PicButton โดยมีรายละเอียดของ Argument ดังนี้ครับ

- **int x** เป็นตำแหน่งที่ต้องการวาดรูปทางแนวแกน x
- **int y** เป็นตำแหน่งที่ต้องการวาดรูปทางแนวแกน y
- **char *Pic** เป็น Address ของข้อมูลรูปภาพที่จะวาด

แนวทางในการออกแบบผมจะอาศัยแนวทางของการสร้างปุ่มกด TextButton เป็นเกณฑ์ ดังนั้นทุก ๆ Method ที่มีใน TextButton จะมีใน Picbutton โดยที่รายละเอียดการทำงานบางส่วนที่จะต้องมีการปรับปรุงดังเช่นที่ผมได้กล่าวไปแล้วข้างต้น ผมจะขอกล่าวเป็นแนวทางคร่าว ๆ ดังนี้ครับ

ในการวาดรูปปุ่มจะเป็นการเรียก PushButton::DrawButtUp() ซึ่งคำสั่งนี้จะทำการวาดรูปปุ่มกด เนื่องจากรูปภาพที่ใช้มีขนาด 24x24 Pixel ผมจึงกำหนดขนาดของปุ่มกดเป็นค่าคงที่ใน Constructor โดยใช้ขนาด 28 x 28 แต่หากท่านผู้อ่านต้องการปุ่มกดขนาดที่โตกว่าที่ผมใช้ก็สามารถทำได้แต่อาจจะต้องมีการกำหนดตำแหน่งของรูปภาพใหม่เพื่อให้เหมาะสมกับขนาดของปุ่มกด เมื่อทำการวาดปุ่มกดแล้วจะทำการเรียกฟังก์ชัน DrawPic() ขึ้นมาทำงานโดยที่ตำแหน่งที่ผมกำหนดไว้ที่พิกัด x,y เป็น SCol+2 ,SRow+3 ส่วนรูปภาพของปุ่มกดขณะถูกกดนั้นก็จะทำการวาดรูปเยื้องไปจากเดิมข้างละ 1 Pixel สำหรับ Method ที่ยังคงการทำงานเดิมไว้จะมี GetButtRelease() และ Pressed()

ส่วนกรณีที่ออกแบบโดยการสืบทอดคุณสมบัติจาก TextButton นั้นการประกาศโครงสร้างของ Object นั้นก็จะมีส่วนที่แตกต่างจากลิสติงที่ 2 เพียงเปลี่ยนการประกาศส่วนหัวของ Class ดังนี้ครับ

```
class PicButton:public TextButton
```

สำหรับรายละเอียดการทำงานนั้นหลาย Method จะยังเหมือนกับการสร้างโดยการรับการสืบทอดจาก Button ที่ต้องเขียนออกมาใหม่มีดังในลิสติงที่ 4

ในการใช้งานออบเจ็กต์ PicButton นี้ต้องมีการส่งผ่าน Argument ต่าง ๆ ดังนี้ครับ

```
PicButton(int Col,int Row,char *title,int
style,unsigned int cmd,int BMsg)
```

- **int Col,Row** เป็นตำแหน่งของปุ่มกดที่จะวาดลงบนหน้าจอตามพิกัด x,y
- **char *title** เป็น Address ของรูปภาพที่จะใช้สำหรับปุ่มกด
- **int style** เป็นสถานะการตอบรับการทำงานซึ่งจะเหมือนกับ TextButton
- **unsigned int cmd** เป็นรหัส Scan Code ของ Keyboard ที่จะใช้สั่งงานแทนการกดเม้าส์
- **int Bmsg** เป็น Message ที่ออบเจ็กต์จะส่งออกมาเมื่อถูกกด

ปรับปรุงโค้ดเก่า

การปรับปรุงโค้ดเก่านั้นหากท่านผู้อ่านเลือกที่จะสืบทอด PicButton จาก TextButton จะมีการแก้ไขโค้ดในแฟ้ม BUTTON.CPP ใน Constructor ของ TextButton ดังนี้ครับ

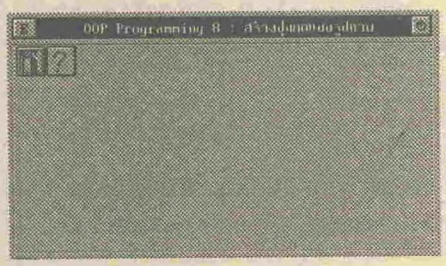
ลิสติงที่ 4

```
PicButton::PicButton(int Col,int Row,char *title,int style,unsigned int cmd,int BMsg):
        TextButton(Col,Row,28,28,NULL,style,cmd,BMsg){
   Title = title;      //point to picture address
   ObjType = PICBUTTON;
   if(Style&VISIBLE)
      DrawButtUp();
}
//*************************************************************************
int PicButton::GetButtRelease(){
return TextButton::GetButtRelease();
}
//*************************************************************************
int PicButton::Pressed(){
return TextButton::Pressed();
}
```

```
TextButton::TextButton(int Col,int Row,int W,int
H,char *title,int style,unsigned int cmd,int
BMsg){
    SCol  = Col;
    SRow  = Row;
    ECol  = SCol + W;
    ERow  = SRow + H;
    Style = style;
    Cmd   = cmd;
    ObjType = TEXTBUTTON;
    RetCmd  = BMsg;
    if(Title!=NULL){  //Revised for OOP8
       Title = new char[strlen(title)+1];
       strcpy(Title,title);
       if(Style&VISIBLE)
        DrawButtUp();
       }
}
```

นอกจากนี้เมื่อสร้างออบเจ็กต์ใหม่ต้องมีการประกาศโครงสร้างของออบเจ็กต์ใหม่ โดยผมได้ประกาศเพิ่มใน OBJECTS.H ถ้าหากเป็นการสืบทอดจาก Button ให้ใช้ลิสติงที่ 2 แต่หากประกาศสร้างโดยการสืบทอดจาก TextButton ยังคงใช้ลิสติงที่ 2 เช่นเดิม เพียงเปลี่ยนที่ส่วนหัวของการประกาศ Class PicButton เป็นดังนี้ครับ

```
class PicButton:public TextButton
```



รูปที่ 2
แสดงผลการทำงาน
ของโปรแกรมตัวอย่าง

โปรแกรมตัวอย่าง

โปรแกรมตัวอย่างในลิสติงที่ 5 เป็นโปรแกรมสั้น ๆ ซึ่งเป็นการทดลองใช้งานปุ่มกดรูปภาพ โดยทั่วไปแล้วจะเหมือนกับปุ่มกดแบบข้อความนั่นเอง โปรแกรมนี้จะสามารถใช้งานได้กับ PicButton ที่สร้างโดยการได้รับการสืบทอดจาก Button หรือ TextButton ครับ (ดูรูปที่ 2)

ส่งท้าย

ในบทความตอนนี้ซอร์สโค้ดทั้งหมดมีให้บริการทางซอฟต์เมล์ โดยท่านผู้อ่านที่สนใจสามารถขอใช้บริการได้ครับ จะประกอบไปด้วยแฟ้มที่จำเป็นทั้งหมดในทุกตอนที่ผ่านมา ผมได้สร้าง Project file ออกเป็น 2 แบบ ซึ่งแยกไว้คนละไดเรคทอรีคือ OOP8_1 และ OOP8_2 โดยแบบแรกคือ OOP8_1.PRJ นั้นจะเป็นการสร้าง PicButton โดยสืบทอดจาก Button และ OOP8_2.PRJ เป็นการสร้าง PicButton โดยสืบทอดจาก TextButton สำหรับผู้อ่านท่านใดมีปัญหาหรือคำแนะนำประการใดสามารถติดต่อผมได้ทาง Intrenet ดังนี้ครับ

Email : ru_boy@poboxes.com
http://www.geocities.com/CollegePark/Union/2029/

สำหรับฉบับนี้ผมขอปิดตอนไว้เพียงเท่านี้ สวัสดีครับ

ลิสติงที่ 5 SAMPLE.CPP

```
 1: #include<graphics.h>
 2: #include"mouobj.h"
 3: #include"stdfon.h"
 4: #include"objects.h"
 5: #include"objinter.h"
 6: #include"ok.inc"
 7: #include"exit.inc"
 8: #include"help.inc"
 9: #define ID_QUIT          10
10: #define ID_HELP          11
11: unsigned int MouEvent;
12: int    MouX,MouY;
13: GUIApp TestApp;
14: WORD   AppMsg;
15: int    Quit = 0;
16: //*************************************************************************
17: void DoMessage(WORD *Msg){
18:    switch(*Msg){
19:      case ID_QUIT  : case CLOSEWIN :
20:             Quit = 1;
21:             break;                                              →
22:      }
```

ลิสติงที่ 5 (ต่อ)

```
23: }
24: //*************************************************************************
25: main(){
26:     int Gd=DETECT,Gm;
27:     initgraph(&Gd,&Gm,"c:\\lang\\bc31\\bgi");
28:     InitMouse();
29:     setrgbpalette(LIGHTGRAY,48,48,48);
30:     setrgbpalette(DARKGRAY+48,35,35,35);
31:     TestApp.AddObj(new Window("OOP Programming 8 : สร้างปุ่มกดแบบรูปภาพ",0,0,639,479,
32:                    VISIBLE|LANG_KEY|SYS_CLOSE,WHITE,BLUE,LIGHTGRAY));
33:     TestApp.AddObj(new PicButton(10,34,ExitPic,VISIBLE|FOCUSED,27,ID_QUIT));
34:     TestApp.AddObj(new PicButton(40,34,HelpPic,VISIBLE,0x0000,ID_HELP));
35:     ShowMouse();
36:     while(!Quit){
37:        GetKbdMouse(&MouEvent,&MouX,&MouY);
38:        AppMsg = TestApp.GetMessage(MouEvent,MouX,MouY);
39:        DoMessage(&AppMsg);
40:        }
41:     HideMouse();
42:     closegraph();
43:     return 0;
44: }
```


ข่าวดี !

นับแต่วันนี้เป็นต้นไป ท่านสามารถสมัครสมาชิกหรือต่ออายุสมาชิกนิตยสารไมโครคอมพิวเตอร์ ได้ที่ร้านซีเอ็ดบุ๊คเซ็นเตอร์ทุกสาขาทั่วประเทศ ดังนี้

กรุงเทพฯ 11 สาขา

ฟอร์จูนทาวน์ เยาฮัน (ชั้นใต้ดิน)
หลักสี่พลาซ่า (ชั้น 3)
เดอะมอลล์ บางกะปิ (ชั้น G)
ฟิวเจอร์พาร์ค รังสิต (ชั้นใต้ดิน)
เดอะมอลล์ บางแค (ชั้น G)
โลตัสซูเปอร์เซ็นเตอร์ พระราม 2 (ชั้น 1)
สีลมคอมเพล็กซ์ (ชั้น 4)
อิมพีเรียลเวิลด์ สำโรง (ชั้นใต้ดิน)
เซ็นทรัลซิตี้ บางนา (ชั้น 3)
เซ็นทรัลพลาซ่า ปิ่นเกล้า (ชั้น 3)
ฟิวเจอร์พาร์ค บางแค (ชั้นใต้ดิน)

ต่างจังหวัด 6 สาขา

โลตัสซูเปอร์เซ็นเตอร์พัทยา ชลบุรี (ชั้น G)
โลตัสซูเปอร์เซ็นเตอร์ สระบุรี (ชั้น G)
โลตัสซูเปอร์เซ็นเตอร์ พิษณุโลก (ชั้น 2)
โลตัสซูเปอร์เซ็นเตอร์ สุพรรณบุรี (ชั้น 2)
โลตัสซูเปอร์เซ็นเตอร์ นครศรีธรรมราช (ชั้น 2)
ร้านเครือข่ายซีเอ็ด ศูนย์หนังสือบุรีรัมย์ บุรีรัมย์

นิตยสารไมโครคอมพิวเตอร์
อัตราค่าสมาชิก **840** บาทต่อปี